We claim:

1. A method for decoupling fetching of an instruction stored in a main program memory from earliest execution of the instruction comprising:

partitioning a program into code segments to be executed by a processor;

establishing corresponding explicit execution instructions for each of the code segments;

fetching a particular code segment;

saving the particular code segment to a storage unit associated with the execution function; and

deferring execution of the particular code segment until initiation of execution is forced by the corresponding explicit execution instruction thereby decoupling execution of the code segments from their fetching.

2. The method of claim 1 further comprising:

repeating the steps of fetching, saving and deferring for all the code segments; and

executing all the code segments in a partition comprising a plurality of processing elements as forced by the corresponding explicit execution instruction.

3. The method of claim 1 further comprising:

preloading local cache storage prior to program execution by fetching a first plurality of instructions from a first code segment and saving the first fetched instructions to a memory serving as a cache for an execute engine; and

fetching a second plurality of instructions from a second code segment and saving those second fetched instruction to a separate location in the cache.

4. The method of claim 3 further comprising:

executing both the first and second code segments in parallel upon encountering their respective explicit execution instructions.

5. A system for decoupling fetching of an instruction stored in a main program memory from earliest execution of the instruction comprising:

means for partitioning a program into code segments to be executed by a processor:

means for establishing corresponding explicit execution instructions for each of the code segments;

means for fetching a particular code segment;

means for saving the particular code segment to a storage unit associated with the execution function; and

means for deferring execution of the particular code segment until initiation of execution is forced by the corresponding explicit execution instruction thereby decoupling execution of the code segments from their fetching.

6. The system of claim 5 wherein the means for fetching, the means for saving and the means for deferring operate for all the code segments, the system further comprising:

means for executing all the code segments in a partition comprising a plurality of processing elements as forced by the corresponding explicit execution instruction.

7. The system of claim 5 further comprising:

means for preloading local cache storage prior to program execution by fetching a first plurality of instructions from a first code segment and saving those first fetched instructions to a memory serving as a cache for an execute engine; and

means for fetching a second plurality of instruction from a second code segment and saving those second fetched instruction to a separate location in the cache.

8. The system of claim 7 further comprising:

means for executing both the first and second code segments in parallel upon encountering their respective explicit execution instructions.

9. A very long instruction word memory system comprising:

a plurality of instruction slots for storing instruction words;

at least one of said plurality of instruction slots having an expanded format over the instruction format required in program storage for storing an expanded instruction having wider instruction width; and

means for loading said at least one expanded format slot with an expanded instruction.

10. The memory system of claim 9 wherein the plurality of instruction slots comprise a store unit instruction slot, a load unit instruction slot, an arithmetic logic unit (ALU) instruction slot, a multiply accumulate unit (MAU) instruction slot, and a data store unit (DSU) instruction slot.

11. The memory system of claim 10 wherein the store unit instruction slots store expanded store instructions including additional bits to extend compute register file addressing, an additional bit to extend address register file addressing, an additional bit to expand an opcode file, or an additional bit to extend a conditional field.

12. The memory system of claim 11 wherein said extended instructions support expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

13. The memory system of claim 10 where the load unit instruction slots store expanded load instructions of a first format for load immediate operations including additional

bits to extend compute register file addressing, a bit to extend address file register addressing, a bit to extend a conditional field or sixteen bits to extend an immediate field.

14. The memory system of claim 13 wherein said extended instructions support expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

15. The memory system of claim 10 where the ALU, MAU and DSU instruction slots store expanded instructions including additional bits in each operand field to extend compute register file addressing, a bit to extend an opcode field, or a bit to extend a data type field.

16. The memory system of claim 15 wherein said extended instructions support expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

17. A very long instruction word (VLIW) instruction memory (VIM) basket (VIMB) system comprising:

an instruction register for storing a load indirect VLIW (LIV) instruction comprising a load mask bit field specifying which slots are to be loaded into the VIMB;

an instruction bit organizer for receiving instructions as data and organizing the bits from the data encoded instructions into proper format for loading into the VIMB; and

the VIMB comprising a plurality of instruction slots having expanded instruction slot width greater than the width of the instruction format required in program storage.

18. A very long instruction word memory system comprising:

a plurality of instructions in a native instruction format of a first bit width and format; and

a very long instruction memory having slots for storing instructions of a second format having a second bit width and format different from that of the native instruction format.

19. The system of claim 18 wherein instructions are stored in a data memory and delivered to the very long instruction memory utilizing a data bus.

20. A very long instruction word memory system comprising:

a plurality of instruction slots for storing instruction words, at least one of said plurality of instruction slots having a compressed format with respect to an instruction format required in the program storage for storing a compressed instruction having narrower instruction width; and

means for loading said at least one compressed format slot with a compressed instruction.

21. A processing apparatus comprising:

a memory for storing a processing apparatus program comprising short instruction words;

an indirect very long instruction word (VLIW) memory comprising a plurality of instruction slots for storing instruction words, at least one of said slot's instruction format sized according to execution function and operand storage capacity and independent of the short instruction word size;

at least one data memory unit storing instruction operands; and

at least two execution units for executing the VLIW's instruction words in response to a short instruction word dispatched from the memory.

22. The processing apparatus of claim 21 wherein the short instruction words comprise K-bits and the slot's instruction format comprises T-bits, wherein $T \neq K$.

23. The processing apparatus of claim 22 wherein the slot's instruction format comprises at least one operand address field of B-bits supporting direct operand addressing.

24. The processing apparatus of claim 23 wherein the data memory unit has a capacity 2^B data values.

25. The processing apparatus of claim 21 wherein the at least one of the two execution units operate on a slot instruction format and directly access operands from the data memory unit for execution.

26. The processing apparatus of claim 21 wherein the two execution units operate as one execution unit when executing two VLIW memory slot instructions as specified by a two slot XV indirect VLIW instruction, and wherein the two execution units operate as one execution unit when executing one VLIW memory slot instruction as specified by a one slot XV indirect VLIW instruction.